ECE 435 – Network Engineering
Lecture 10

Vince Weaver

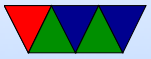
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

4 October 2018

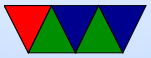
Announcements

- HW#4 was due
- HW#5 will be posted. You'll have 2 weeks due to midterm/fall break

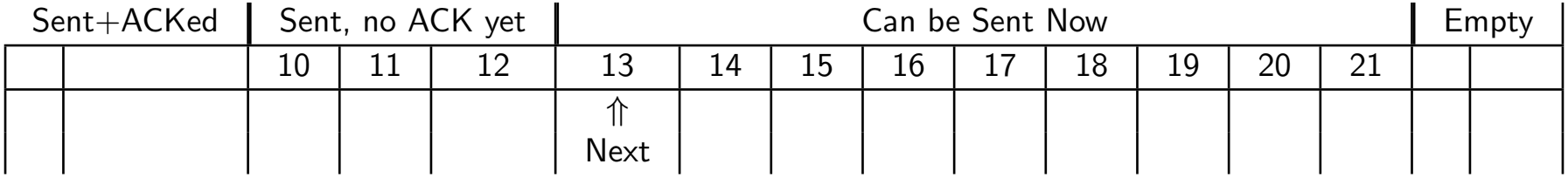


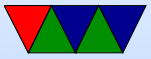
Flow Control

- How much data can be sent before receiving an ACK
- Extreme – just 1 byte. Inefficient (overhead). Also modern systems, a fibre line coast to coast a long time to ACK packet
- How does operating system / TCP stack keep track of this
  - Sliding Window Protocol
  - Circular buffer (see figure)
  - Size of the sliding window: how many outstanding



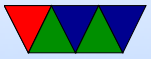
there can be. Once it gets ACKed, can slide window. grow/shrink size of window.





Receiver Window

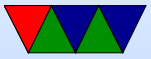
- Receiver Window (RWND)
- example
  - Receiver has 4k buffer
  - Sender does 2k write (2k/SEQ=0)
  - Receiver sends back ACK=2k, WIN=2048 (can take up to 2k)
  - Application sends 2k (2k, SEQ=2k)
  - If it is full, receiver might send ACK=4k, WIN=0
  - Later once buffer clears up a bit (application reads 2k



maybe) sends ACK=4096, WIN=2k

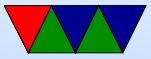
- Sender then sends some more

- When waiting on a WIN=0 can send two things, URG to kill the connection, or a 1-byte packet to have retransmit window and next byte expected (in case the ack restarting was lost, otherwise deadlock)



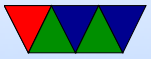
Window Management / Flow Control

- Senders do not have to transmit incoming data immediately
- Receivers do not have to ACK immediately
- Can have a lot of overhead to send 40byte packet for one byte payload
- Senders can buffer data, for example if know window is 4k can wait until they have 4k. Can help performance.
- For example, typing at keyboard on telnet/ssh might send when an editor. Press key, send a packet. Get



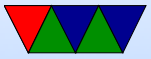
ACK. Then when read, another ACK updating window size. Then finally draw char on screen, send packet with that. 4 packets for one keypress

- One way to help is avoid window updates for up to 500ms in hope they can tag along with a real outgoing packet



Silly Window Syndrome

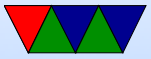
- Worst case, read one byte at a time, and then huge packet overhead for just one byte. Can be sender or receiver.
- Solution on sender end
  - Nagel's algorithm (John Nagel, 1984) RFC896
  - When only sending one byte at a time, send one packet, but buffer the rest until the outstanding data is ACKed
  - Also take into account window size.



- Widely used, can be bad for things like X window forwarding as mouse movements bunched together.
- Interacts poorly with Delayed ACKs
- TCP_NODELAY option disables.

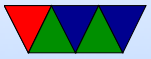
- Solution on receiver end
  - Clark's solution
  - application reading out the bytes one at a time.
  - If receiving small amounts, close window until buffer half empty and the open again.
- Another – Delayed ACK

  delay the ACK so the other side has time to queue up



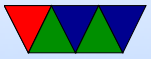
data

Can't delay by more than 500ms though

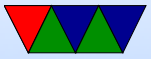


TCP Congestion Control

- Fast network feeding small receiver (flow control?)
- Slow network feeding big receiver (problem on sending side, lose packets, congestion control)
- Two windows, receiver window (RWND) looked at previously, and congestion window (CWND) (kept locally)
- Amount can send is the minimum of the two windows
- Rough idea
  - Congestion window set to max segment size

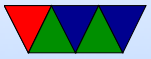


- Send one max segment, arm timer
- If ACK received before timer goes off, good. Double the size.
- Repeat, exponential growth. Called "slow start"
- "internet (?)" has a limit where it stops exponential and moves to linear growth
- eventually hit receiver window size and stop



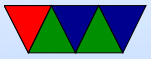
TCP Congestion Control – History

- Originally no congestion control, not needed (not that much traffic)
- 1980s Internet facing "congestion collapse" – would send as fast as possible, packets would be dropped, hosts retransmit, even more congestion
- Usually assume corrupted packets are rare, at least on wired. Wireless. More lost packets, so shouldn't slow down but maybe try harder in congestion



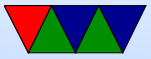
TCP Tahoe

- TCP Tahoe (v2) (BSD 4.2 1988) added congestion avoidance, fast retransmit (Van Jacobsen)
- Adds CWND (congestion windw) to control data sent in one round-trip-time with a MWND maximum-window size.
- Slow-start
  - Quickly probe bandwidth with a few rounds.
  - CWND set to 1
  - CWND increased exponentially by doubling each time



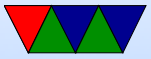
gets ACK. until hits ssthresh (Slow-start threshold)

- Congestion avoidance
  - Slowly probe bandwidth but rapidly respond to congestion
  - Enter once slow-start hits ssthresh
  - Only increase linearly
  - AIMD (additive increase multiple decrease)
  - If get three duplicate ACKS in a row, assume packet loss, resend. Drop ssthresh to half and start slow-start again
- Fast retransmit

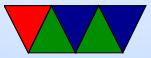


- re-transmit lost packets immediately, no wait for timer.
- If get three duplicate ACKS in a row, assume packet loss, resend. Drop ssthresh to half and start slow-start again

- Retransmission timeout – halve sshthreash and restart slow-start
- When to retransmit
  - If packet lost, then will receive duplicate ACK on next transmission
  - If get three identical ACKs, probably means packet lost, resend

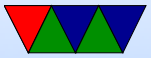


- Why not two? Because if packets arrive out of order can also cause duplicate ACK



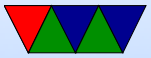
TCP Reno

- Adds fast-recovery state, to recover faster if packet loss
- set sshthread to cwnd+3 because of triple ack



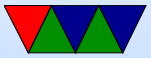
TCP Vegas

- Delays measured for every packet?
- Proposed, not widely used?



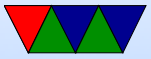
TCP New Reno

- RFC 6582
- Widely used today?



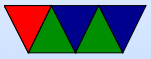
TCP Other

- TCP Hybla
- TCP BIC
- TCP CUBIC
- Agile-SD TCP
- TCP Westwood+
- Compound TCP (Microsoft)
- TCP Propotional Rate Reduction (current Linux)
- TCP BBR (Bottleneck BW and Roun-trip) Google
- Others (many many)



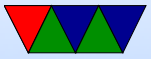
TCP Header – Options

- One-byte
  - End of option – end of all options. Only one allowed (not always needed?)
  - No operation – for padding
- Multi-byte
  - MSS maximum segment size (only in initial SYN packet)
    Byte1: 2, Byte 2: len=4, max size=2 bytes
  - RFC1323 –PAWS, window scaling factor, specify



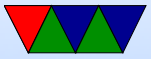
larger transfer size as on long-latency high-bandwidth connections can end up idle a lot waiting for ACK
Scales the window size by value
Byte 1: 3, Byte 2: Len=3, Scale factor=1 byte

- Timestamp: used to calculate round-trip time. Leaks info? Send along timestamp when you send, other side keeps that and returns it (as echo) with the relevant ACK.
  10 bytes long Byte 1: 8, Byte 2: Len=10, Timestamp=4 bytes, Echo=4 bytes
- RFC1106 allows selective resend – if lost packet in



long stream, instead of sending all, just resend missing

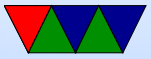
- Fast connections sequence can wrap quickly.



Security Issues

- SYN Flood attack – Denial of service – spoof IP address, send lots of spurious SYNs followed by ACK, tie up lots of resources
  Spoof, because responds to wrong address which just ignores. Causes lots of half-open connections
  One solution is SYN cookies – (pick special sequence that allow throwing out connection info but able to reconstruct if an ACK comes back).

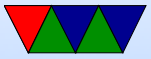
- Connection hijacking – guess a proper sequence number



and forge a packet that looks like it should be next. If you can also take down the real IP (DOS?) can take over the connection Helps to have good random sequence numbers

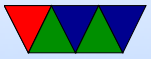
- TCP veto – inject packet with sequence and payload of next expected. That way when the real actual next one comes in, it will be silently dropped as a duplicate

- NMAP port scanning – send packets and find if connections are open, determine host system. Christmas Tree packets



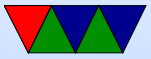
Find out host? options supported, sequence numbers. Also can find out uptime of system as timestamps from extension usually aren't reset each time.

- Martian packets – packets with a source of a reserved network found on routed internet
- UDP broadcast storm/amplification attack – broadcast bad packet to 10000 machines, all reply at once with error



Making things faster

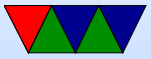
- Offload engines



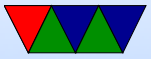
Proposed Replacements

- T/TCP – Transactional TCP.
  To send one small message (w/o UDP) can require up to 9 packets. (handshake, data transfer, shutdown)
  Instead, in the initial packet put SYN, DATA, AND FIN.
  The small message done in 3 packets.

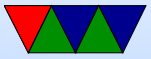
- SCTP – stream control transmission protocol. More complex
  four-way handshake (why? prevent SYN flood attacks)



- QUIC – from google
  Over UDP for now (NATs won't route protocols they don't know)
  Used by Youtube, Google, etc. with Chromium
  Head of Line problem with TCP
  Single-way handshake if version match, otherwise has to negotiate.
  Encrypted
  Once encrypted connection set up once, assumed still there and so sends single HELLO packet followed by data.

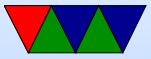


Sends redundancy in packets which can be used with XOR to reconstruct missing packets. (but shown not to help much?)

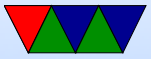


Sample Packets

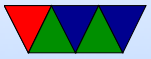
```
sudo tcpdump -XX -i eth0 tcp port 31337
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
15:00:07.507769 IP 192.168.8.38.57962 > 192.168.8.138.31337: Flags [S],
seq 2755900677, win 29200,
options [mss 1460,sackOK,TS val 2941869143 ecr 0,nop,wscale 7], length 0
0x0000:  b827 ebf8 7742 0050 b647 1cde 0800 4510  .'..wB.P.G....E.
0x0010:  003c c79f 4000 4006 e10b c0a8 0826 c0a8  .<..@.@......&..
0x0020:  088a e26a 7a69 a443 b505 0000 0000 a002  ...jzi.C........
0x0030:  7210 8220 0000 0204 05b4 0402 080a af59  r..............Y
0x0040:  5c57 0000 0000 0103 0307                 \W........
15:00:07.507901 IP 192.168.8.138.31337 > 192.168.8.38.57962: Flags [S.],
seq 3080782625, ack 2755900678, win 28960,
options [mss 1460,sackOK,TS val 3332076052 ecr 2941869143,nop,wscale 7],
length 0
0x0000:  0050 b647 1cde b827 ebf8 7742 0800 4500  .P.G...'..wB..E.
0x0010:  003c 0000 4000 4006 a8bb c0a8 088a c0a8  .<..@.@.........
0x0020:  0826 7a69 e26a b7a1 0321 a443 b506 a012  .&zi.j...!.C....
0x0030:  7120 922f 0000 0204 05b4 0402 080a c69b  q../............
0x0040:  7214 af59 5c57 0103 0307                 r..Y\W....
15:00:07.508278 IP 192.168.8.38.57962 > 192.168.8.138.31337: Flags [.],
ack 1, win 229, options [nop,nop,TS val 2941869143 ecr 3332076052], length 0
```



```
0x0000:  b827 ebf8 7742 0050 b647 1cde 0800 4510  .'..wB.P.G....E.
0x0010:  0034 c7a0 4000 4006 e112 c0a8 0826 c0a8  .4..@.@......&..
0x0020:  088a e26a 7a69 a443 b506 b7a1 0322 8010  ...jzi.C.....".. 
0x0030:  00e5 2e94 0000 0101 080a af59 5c57 c69b  ...........Y\W..
0x0040:  7214                                     r.
15:00:09.037573 IP 192.168.8.38.57962 > 192.168.8.138.31337: Flags [P.],
seq 1:6, ack 1, win 229,
options [nop,nop,TS val 2941870672 ecr 3332076052],
 length 5
0x0000:  b827 ebf8 7742 0050 b647 1cde 0800 4510  .'..wB.P.G....E.
0x0010:  0039 c7a1 4000 4006 e10c c0a8 0826 c0a8  .9..@.@......&..
0x0020:  088a e26a 7a69 a443 b506 b7a1 0322 8018  ...jzi.C.....".. 
0x0030:  00e5 3d1b 0000 0101 080a af59 6250 c69b  ..=........YbP..
0x0040:  7214 6865 790d 0a                        r.hey..
15:00:09.037634 IP 192.168.8.138.31337 > 192.168.8.38.57962: Flags [.],
ack 6, win 227, options [nop,nop,TS val 3332077582 ecr 2941870672], length 0
0x0000:  0050 b647 1cde b827 ebf8 7742 0800 4500  .P.G...'..wB..E.
0x0010:  0034 3e1c 4000 4006 6aa7 c0a8 088a c0a8  .4>.@.@.j.......
0x0020:  0826 7a69 e26a b7a1 0322 a443 b50b 8010  .&zi.j..."..C....
0x0030:  00e3 9227 0000 0101 080a c69b 780e af59  ...'........x..Y
0x0040:  6250                                     bP
15:00:09.039363 IP 192.168.8.138.31337 > 192.168.8.38.57962: Flags [P.],
 seq 1:6, ack 6, win 227, options [nop,nop,TS val 3332077584 ecr 2941870672],
 length 5
```



```
0x0000:  0050 b647 1cde b827 ebf8 7742 0800 4500  .P.G...'..wB..E.
0x0010:  0039 3e1d 4000 4006 6aa1 c0a8 088a c0a8  .9>.@.@.j.......
0x0020:  0826 7a69 e26a b7a1 0322 a443 b50b 8018  .&zi.j..."".C....
0x0030:  00e3 922c 0000 0101 080a c69b 7810 af59  ...,........x..Y
0x0040:  6250 4845 590d 0a                        bPHEY..
15:00:09.039650 IP 192.168.8.38.57962 > 192.168.8.138.31337: Flags [.],
ack 6, win 229, options [nop,nop,TS val 2941870674 ecr 3332077584], length 0
0x0000:  b827 ebf8 7742 0050 b647 1cde 0800 4510  .'..wB.P.G....E.
0x0010:  0034 c7a2 4000 4006 e110 c0a8 0826 c0a8  .4..@.@......&..
0x0020:  088a e26a 7a69 a443 b50b b7a1 0327 8010  ...jzi.C.....'..
0x0030:  00e5 2293 0000 0101 080a af59 6252 c69b  ..".........YbR..
0x0040:  7810                                     x.
15:00:12.469960 IP 192.168.8.38.57962 > 192.168.8.138.31337: Flags [F.],
 seq 6, ack 6, win 229, options [nop,nop,TS val 2941874105 ecr 3332077584],
 length 0
0x0000:  b827 ebf8 7742 0050 b647 1cde 0800 4510  .'..wB.P.G....E.
0x0010:  0034 c7a3 4000 4006 e10f c0a8 0826 c0a8  .4..@.@......&..
0x0020:  088a e26a 7a69 a443 b50b b7a1 0327 8011  ...jzi.C.....'..
0x0030:  00e5 152b 0000 0101 080a af59 6fb9 c69b  ...+.......Yo...
0x0040:  7810                                     x.
15:00:12.471639 IP 192.168.8.138.31337 > 192.168.8.38.57962: Flags [F.],
 seq 6, ack 7, win 227, options [nop,nop,TS val 3332081016 ecr 2941874105],
 length 0
0x0000:  0050 b647 1cde b827 ebf8 7742 0800 4500  .P.G...'..wB..E.
```



```
0x0010:  0034 3e1e 4000 4006 6aa5 c0a8 088a c0a8  .4>.@.@.j.......
0x0020:  0826 7a69 e26a b7a1 0327 a443 b50c 8011  .&zi.j...'.C....
0x0030:  00e3 9227 0000 0101 080a c69b 8578 af59  ...'.........x.Y
0x0040:  6fb9                                     o.
15:00:12.471921 IP 192.168.8.38.57962 > 192.168.8.138.31337: Flags [.],
 ack 7, win 229, options [nop,nop,TS val 2941874107 ecr 3332081016], length 0
0x0000:  b827 ebf8 7742 0050 b647 1cde 0800 4510  .'..wB.P.G....E.
0x0010:  0034 c7a4 4000 4006 e10e c0a8 0826 c0a8  .4..@.@......&..
0x0020:  088a e26a 7a69 a443 b50c b7a1 0328 8010  ...jzi.C.....(..
0x0030:  00e5 07c0 0000 0101 080a af59 6fbb c69b  ...........Yo...
0x0040:  8578                                     .x
```